МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО

Факультет автоматизированных и информационных систем

Кафедра «Информатика»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 3

по дисциплине «РПИ»

на тему: «NodeJs EventLoop»

Выполнил: студент гр. ИП-32

Суховенко Э.С.

Принял: преподаватель

Процкая М.А.

Гомель 2022

**Цель работы:** изучить основные понятия EventLoop, возможность по работе с событиями в NodeJs

**Задание**

Задание 1 Написать скрипт, который наглядно будет демонстрировать работу каждой из фаз EventLoop:

1.1 Создать три таймера SetTimeOut с разным временем срабатывания. При срабатывании таймера выводить время, прошедшее с момента запуска таймера.

1.2 Применить операции ввода-вывода для записи и чтения файла. После окончания операции ввода-вывода выводить время, прошедшее с момента начала операции.

1.3 Добавить вызов SetImmediate. Вывести время, прошедшее с момента вызова.

Задание 2 Зарегистрировать в системе пользовательское событие. Сгенерировать и обработать пользовательское событие с выводом информации о событии.

Задание 3 Промоделировать ситуацию, в которой Вы являетесь продавцом товара (описание товара в лабораторной №2). На обновление списка товара подписать нескольких клиентов, которым будет приходить сообщение при добавлении нового товара или модификации старого. Клиенты должны логгировать информацию от продавца в своих журналах в виде файла с занесением времени.

**Код программы**

// TIMERS ­\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// index.ts

import { *task1* } from './tasks/task1'  
import { *task2* } from './tasks/task2'  
import { *task3* } from './tasks/task3'  
  
const main = async () => {  
 const tasks = [*task1*, *task2*, *task3*]  
 for (let i = 0; i < tasks.length; ++i) {  
 console.log(`---TASK [${i + 1}]---`)  
 await tasks[i]()  
 }  
}  
  
main()

// asyncUtils.ts

import fs from 'fs'  
import ErrnoException = NodeJS.ErrnoException  
  
export const *asyncTimeout* = async (callback: () => void, time: number) => {  
 return new Promise((resolve: (value: unknown) => void) => *setTimeout*(() => {  
 callback()  
 resolve(null)  
 }, time))  
}  
  
export const *asyncImmediate* = async (callback: () => void) => {  
 return new Promise((resolve: (value: unknown) => void) => *setImmediate*(() => {  
 callback()  
 resolve(null)  
 }))  
}  
  
export const *asyncReadFile* = async (filePath: string) => {  
 return new Promise((resolve: (value: string) => void) => {  
 fs.*readFile*(filePath, (err: ErrnoException, data: Buffer) => {  
 resolve(data.toString())  
 })  
 })  
}

// task1.ts

import { *asyncTimeout* } from './asyncUtils'  
  
export const *task1* = async () => {  
 console.time('timer1')  
 console.time('timer2')  
 console.time('timer3')  
  
 await Promise.all([  
 *asyncTimeout*(() => console.timeEnd('timer1'), 100),  
 *asyncTimeout*(() => console.timeEnd('timer2'), 200),  
 *asyncTimeout*(() => console.timeEnd('timer3'), 300),  
 ])  
}

// task2.ts

import { *asyncReadFile* } from './asyncUtils'  
  
export const *task2* = async () => {  
 *// fs.writeFileSync('./res/out.json', JSON.stringify(Array(15).fill(null).map((x, i) => i), null, ' '))* console.time('file reading')  
 console.log(JSON.parse(await *asyncReadFile*('./res/out.json')))  
 console.timeEnd('file reading')  
}

// task3.ts

import { *asyncImmediate* } from './asyncUtils'  
  
export const *task3* = async () => {  
 console.time('immediate')  
 await *asyncImmediate*(() => console.timeEnd('immediate'))  
}

// EVENTS \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// index.ts

import EventEmitter from 'events'  
import \* as fs from 'fs'  
  
class MyEmitter extends EventEmitter {}  
  
const myEmitter = new MyEmitter()  
  
myEmitter.on('event', (data: unknown[]) => {  
 const currentLogs = JSON.parse(fs.*readFileSync*('./out/out.json').toString() || '[]')  
 fs.*writeFileSync*('./out/out.json', JSON.stringify([  
 ...currentLogs,  
 {  
 data: JSON.stringify(data),  
 time: new Date().toUTCString(),  
 author: 'Eduard',  
 },  
 ], null, ' '))  
})  
  
fs.*truncateSync*('./out/out.json')  
console.log('Program started, out.json was cleared.')  
myEmitter.emit('event', [0, 0, 0, 0, 0])  
*setTimeout*(() => myEmitter.emit('event', [1]), 1000)  
*setTimeout*(() => myEmitter.emit('event', [1, 2]), 2000)  
*setTimeout*(() => myEmitter.emit('event', [1, 2, 3]), 3000)

// out.json

[  
 {  
 "data": "[0,0,0,0,0]",  
 "time": "Sun, 06 Mar 2022 19:57:28 GMT",  
 "author": "Eduard"  
 },  
 {  
 "data": "[1]",  
 "time": "Sun, 06 Mar 2022 19:57:29 GMT",  
 "author": "Eduard"  
 },  
 {  
 "data": "[1,2]",  
 "time": "Sun, 06 Mar 2022 19:57:30 GMT",  
 "author": "Eduard"  
 },  
 {  
 "data": "[1,2,3]",  
 "time": "Sun, 06 Mar 2022 19:57:31 GMT",  
 "author": "Eduard"  
 }  
]

// STORE \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// index.ts

import { Client } from './store/client'  
import { Store } from './store/store'  
import { OperatingSystem } from './constants/types'  
  
const generateFilePath = (index: number, baseFilePath = './out/') => (`${baseFilePath}out${index}.json`)  
  
const total = 3  
const clients: Client[] = Array(total).fill(null).map((client, index) => new Client(generateFilePath(index + 1)))  
const store: Store = new Store()  
  
clients.forEach(client => {  
 store.subscribeClient(client)  
 client.clearLogs()  
})  
  
store.notifyClients() *// 1. android*store.setOperatingSystem(OperatingSystem.*ios*) *// 2. ios*store.unsubscribeClient(clients[2]) *// unsubscribe second client*store.setOperatingSystem(OperatingSystem.*windows*) *// 3. windows*store.unsubscribeClient(clients[1]) *// unsubscribe third client*store.setOperatingSystem(OperatingSystem.*android*) *// 4. android*

// constants/interfaces.ts

import { OperatingSystem, Tablet } from './types'  
  
export type TabletsInfoListener = (tabletInfo: Tablet) => void  
  
export interface IClient {  
 updateTabletsInfo: TabletsInfoListener  
 clearLogs: () => void  
}  
  
export interface IStore {  
 notifyClients: () => void  
 subscribeClient: (client: IClient) => void  
 unsubscribeClient: (client: IClient) => void  
  
 setOperatingSystem: (operatingSystem: OperatingSystem) => void  
}

// constants/types.ts

export enum Diagonal {  
 *\_2560x1440* = '2560x1440',  
 *\_1920x1080* = '1920x1080',  
 *\_1280x720* = '1280x720',  
 *\_1600x900* = '1600x900',  
}  
  
export enum CPU {  
 *a11* = 'a11',  
 *snapdragon600* = 'snapdragon600',  
 *mediatek1200* = 'mediatek1200',  
}  
  
export enum Resolution {  
 *five* = '5.0',  
 *six* = '6.0',  
 *seven* = '7.0',  
}  
  
export enum OperatingSystem {  
 *android* = 'android',  
 *ios* = 'ios',  
 *windows* = 'windows',  
}  
  
export enum RandomAccessMemory {  
 *small* = '1',  
 *large* = '4',  
 *superLarge* = '8',  
}  
  
export enum LocalMemory {  
 *small* = '16',  
 *large* = '32',  
 *superLarge* = '64',  
}  
  
export type Tablet = {  
 diagonal: Diagonal  
 CPU: CPU  
 resolution: Resolution  
 operatingSystem: OperatingSystem,  
 randomAccessMemory: RandomAccessMemory,  
 localMemory: LocalMemory,  
}

// store/client.ts

import { IClient } from '../constants/interfaces'  
import { Tablet } from '../constants/types'  
import \* as fs from 'fs'  
  
export class Client implements IClient {  
 filePath: string  
  
 constructor(filePath: string) {  
 this.filePath = filePath  
 }  
  
 updateTabletsInfo = (tabletInfo: Tablet): void => {  
 const previousInfo = JSON.parse(fs.*readFileSync*(this.filePath).toString() || '[]')  
 const newInfo = [...previousInfo, {  
 tabletInfo,  
 time: new Date().toUTCString(),  
 }]  
 fs.*writeFileSync*(this.filePath, JSON.stringify(newInfo, null, ' '))  
 }  
  
 clearLogs = (): void => {  
 fs.*truncateSync*(this.filePath)  
 }  
}

// store/store.ts

import { CPU, Diagonal, LocalMemory, OperatingSystem, RandomAccessMemory, Resolution, Tablet } from '../constants/types'  
import { IClient, IStore, TabletsInfoListener } from '../constants/interfaces'  
  
export class Store implements IStore {  
 tabletInfo: Tablet  
  
 subscribers: TabletsInfoListener[]  
  
 constructor() {  
 this.tabletInfo = {  
 diagonal: Diagonal.*\_1280x720*,  
 CPU: CPU.*snapdragon600*,  
 resolution: Resolution.*seven*,  
 operatingSystem: OperatingSystem.*android*,  
 randomAccessMemory: RandomAccessMemory.*large*,  
 localMemory: LocalMemory.*large*,  
 }  
 this.subscribers = []  
 }  
  
 setOperatingSystem = (operatingSystem: OperatingSystem): void => {  
 this.tabletInfo = {  
 ...this.tabletInfo,  
 operatingSystem,  
 }  
 this.notifyClients()  
 }  
  
 subscribeClient = (client: IClient): void => {  
 if (this.subscribers.includes(client.updateTabletsInfo)) return  
 this.subscribers.push(client.updateTabletsInfo)  
 }  
  
 unsubscribeClient = (client: IClient): void => {  
 if (!this.subscribers.includes(client.updateTabletsInfo)) return  
 this.subscribers = this.subscribers.filter(subscriber => subscriber != client.updateTabletsInfo)  
 }  
  
 notifyClients = (): void => {  
 this.subscribers.forEach(subscriber => subscriber(this.tabletInfo))  
 }  
}

**Результат**

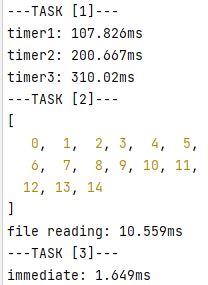


Рисунок 1

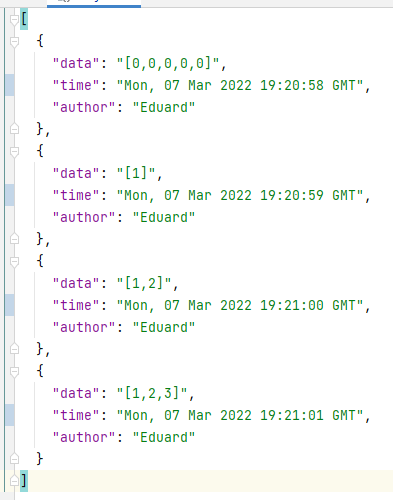




Рисунок 2

 Client1 Client2 Client3



Рисунок 3

**Вывод**: изучил основные понятия EventLoop, возможность по работе с событиями в NodeJs